tools/aapt/Resource.cpp

parsePackage

解析AndroidManifest.xml

将AndroidManifest.xml进行flatten

## AaptAssets类

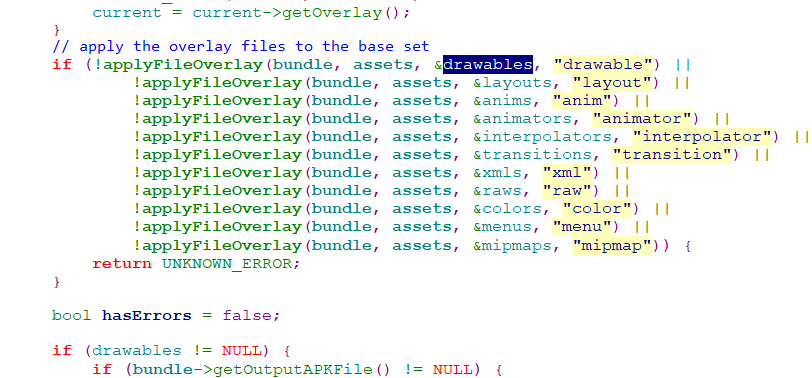
存储文件，不包含values

AssetManager mIncludedAssets，包含通过-i指定的已编译好的包，如android\sdk\platforms\android-27\android.jar

Vector<sp<AaptDir> > mResDirs，收集完成文件之后的资源目录，resType-》AaptDir，resType记录在AaptDir的mLeaf字段中

sp<AaptAssets> mOverlay 制定了多个res目录，第一个之后的都是overlay

applyFileOverlay用overlay的覆盖base中的，需要AaptGroupEntry一样，即配置信息一样。如果base中不错在，直接添加



**compileResourceFile**编译values目录下的资源

调用parseXMLResource，然后

## AaptFile

getSourceFile()文件路径

## Package类

sp<AaptFile> mTypeStringsData; 类型StringPool转换成的AaptFile

ResStringPool mTypeStrings; 类型字符串池，如drawable，layout

DefaultKeyedVector<String16, uint32\_t> mTypeStringsMapping; 类型字符串和typeId(在ResStringPool中的索引)的映射

sp<AaptFile> mKeyStringsData; key StringPool转换成的AaptFile

ResStringPool mKeyStrings; 资源名字符串资源池，R.type.xxx,xxx这部分

DefaultKeyedVector<String16, uint32\_t> mKeyStringsMapping; 键值字符串和entryId(在ResStringPool中的索引)的映射

typeString 类型字符串池，如drawable，layout

keyString

valueString?

ResourceTable::Entry::prepareFlatten

## Type类

DefaultKeyedVector<String16, Public> mPublic public.xml中的信息

## Entry类

**mNameIndex** entry's name string in the key pool，在Package的mTypeStrings中的索引

## StringPool类

**字符串vector，同一字符串只存在一个，entry 中的indices为该entry在mEntryArray中的index**

**Vector<entry> mEntries;**

**存储字符串在mEntries中的索引，如果同一字符串add了多次，在mEntryArray中就会存在多个元素，但是值相同，对应与mEntries中的同一个值**

**Vector<size\_t> mEntryArray;**

**元素与mEntryArray对应，表示对应的entry的style，一个entry会有多个style**

**Vector<entry\_style> mEntryStyleArray;**

**字符串，ssize\_t位该字符串对应于mEntryArray的索引**

**DefaultKeyedVector<String16, ssize\_t> mValues;**

**add函数返回的是字符串对应于mEntryArray的索引，mEntryArray[pos]**

**StringPool::writeStringBlock**

**header->header.type = htods(RES\_STRING\_POOL\_TYPE);**

**header->header.headerSize = htods(sizeof(\*header));**

**header->header.size = htodl(pool->getSize());**

**header->stringCount = htodl(ENTRIES);**

**header->styleCount = htodl(STYLES);**

**if (mUTF8) {**

**header->flags |= htodl(ResStringPool\_header::UTF8\_FLAG);**

**}**

**header->stringsStart = htodl(preSize);**

**header->stylesStart = htodl(STYLES > 0 ? (preSize+strPos) : 0);**

**// Write string index array**

**// Write style index array**

**// Write strings**

**// Write styles**

**attr.nameResId**

**attr.namePoolIdx**

**resids**

**XMLNode::flatten**

## ResTable类

ResTable::addInternal解析resources.arsc

**slurpFromArgs解析参数，收集文件**

****

## buildResources

### parsePackage

解析AndroidManifest.xml

获取包名，填入AaptAsset的mPackage中

**ResourceTable table(bundle, String16(assets->getPackage()), packageType);**

构造ResourceTable，设置packageId

App和AppFeature的packageId为0x7f

System的packageId为0x01

SharedLibrary的packageId为0x00

### ResourceTable::addIncludedResources

添加依赖资源包，比如系统资源包android.jar，里面包含系统资源的定义

### collect\_files(assets, resources)

将所有的资源文件保存到resources，然后保存到

sp<ResourceTypeSet> drawables;

sp<ResourceTypeSet> layouts;

sp<ResourceTypeSet> anims;

sp<ResourceTypeSet> animators;

sp<ResourceTypeSet> interpolators;

sp<ResourceTypeSet> transitions;

sp<ResourceTypeSet> xmls;

sp<ResourceTypeSet> raws;

sp<ResourceTypeSet> colors;

sp<ResourceTypeSet> menus;

sp<ResourceTypeSet> mipmaps;

### applyFileOverlay

### preProcessImages

<https://zh.wikipedia.org/wiki/PNG#%E6%96%87%E4%BB%B6%E7%BB%93%E6%9E%84>

处理.9.png文件

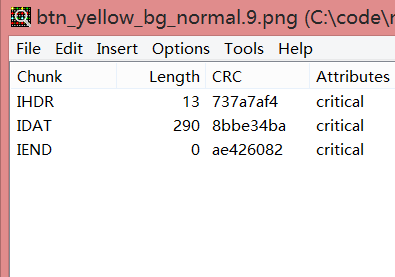
PNG图像格式文件由一个8字节的PNG文件标识（file signature）域和3个以上的后续数据块chunk（IHDR、IDAT、IEND）组成

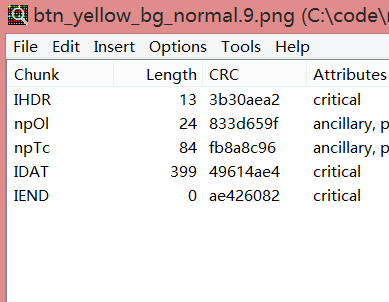
PNG定义了两种类型的数据块：一种是PNG文件必须包含、读写软件也都必须要支持的关键块（critical chunk）；另一种叫做辅助块（ancillary chunks）

关键数据块中有4个标准数据块：

* 文件头数据块IHDR（header chunk）：包含有图像基本信息，作为第一个数据块出现并只出现一次。
* 调色板数据块PLTE（palette chunk）：必须放在图像数据块之前。
* 图像数据块IDAT（image data chunk）：存储实际图像数据。PNG数据允许包含多个连续的图像数据块。
* 图像结束数据IEND（image trailer chunk）：放在文件尾部，表示PNG数据流结束。

png\_set\_unknown\_chunks设置9patch的chunk数据





### makeFileResources

处理非values资源

遍历ResourceTypeSet，将收集到的resource分别存入AaptAsset和ResourceTable

存储values和文件路径信息，对应resources.arsc

ResourceTable，package，type，entry，item，value保存文件路径

### compileResourceFile

编译res下的values资源

**parseXMLResource** 将一个xml资源文件编译为二进制xml

**XMLNode::flatten**

**collect\_attr\_strings**收集有id的attr name，将name加到StringPool strings，将id加到Vector resids，以在StringPool中的index作为id在resids中的index，即位置对应

**collect\_strings** 把xml节点名，属性名，属性值都加入到StringPool中

collect\_attr\_strings

**createStringBlock**序列化stringpool

**flatten\_node**节点名，属性名，属性值全部替换为Stringpool中的index

**ResXMLTree.setTo**将二进制xml写入ResXMLTree, ResXMLTree代表最终的二进制xml

compileAttribute函数

parseAndAddBag函数

context.obtainStyledAttributes

生成ResourceTable下的Package，Type，Entry，Bag，Item

TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.***RoundedImageView***, defStyle, 0);  
  
**int** index = a.getInt(R.styleable.***RoundedImageView\_android\_scaleType***, -1);  
**if** (index >= 0) {  
 setScaleType(***SCALE\_TYPES***[index]);  
} **else** {  
 *// default scaletype to FIT\_CENTER* setScaleType(ScaleType.***FIT\_CENTER***);  
}  
  
**cornerRadius** = a.getDimensionPixelSize(R.styleable.***RoundedImageView\_riv\_corner\_radius***, -1);  
**borderWidth** = a.getDimensionPixelSize(R.styleable.***RoundedImageView\_riv\_border\_width***, -1);

### ResourceTable::assignResourceIds

按资源出现的顺序为资源分配id(结合public中指定的值)

给type，entry，bag分配id

type从1开始递增

entry从0开始递增，即索引

Entry::generateAttributes给id类型的bag分配entry，对于id类型的资源，value为"false"

Type::applyPublicEntryOrder根据public.xml指定的id值更新id值

Entry::assignResourceIds给bag资源设置id

### compileXmlFile

### postProcessImage

### ResourceTable::addSymbols

**生成R.java**

### ResourceTable::flatten

**生成resources.arsc**

**ResTable::stringToValue**